ROSALES, Angel Abegail B.

BSIT 3-5

Activity 4: Manipulating and Merging Data

Instructions

- Manipulate and combine datasets using R. Write and test your R code for each question. Provide explanations for your steps and results.

1. Multi-Column Merge: Create two data frames:

   df1: Columns ID, Name, and Age.

   df2: Columns ID, Score, and Subject.
   Write R code to perform an inner join on ID and explain the resulting data frame structure.

```
> # Make the data frames
> df1 <- data.frame(ID = c(1, 2, 3),
+                   Name = c("Alice", "Bob", "Charlie"),
+                   Age = c(23, 35, 42))
> df2 <- data.frame(ID = c(1, 2, 4),
+                   Score = c(85, 90, 78),
+                   Subject = c("Math", "Science", "History"))
>
> # Do an inner join on ID
> merged_df <- merge(df1, df2, by = "ID")
>
> print(merged_df)
  ID  Name Age Score Subject
1  1 Alice  23    85    Math
2  2   Bob  35    90 Science
```

   The inner join combines rows from both data frames where the ID exists in both `df1` and `df2`. The result includes columns for ID, Name, Age, Score, and Subject, with only the rows that have matching ID values in both data frames appearing in the final result.

2. Add a column to the merged data frame from Task 1 that dynamically categorizes Age into three groups:

   "Young": Age <= 25

   "Middle-aged": Age > 25 and <= 40

   "Senior": Age > 40

```
> # Add Age group column
> merged_df$AgeGroup <- cut(merged_df$Age,
+                           breaks = c(-Inf, 25, 40, Inf),
+                           labels = c("Young", "Middle-aged", "Senior"))
>
> # View the updated data frame
> print(merged_df)
  ID  Name Age Score Subject    AgeGroup
1  1 Alice  23    85    Math       Young
2  2   Bob  35    90 Science Middle-aged
>
```

   The `cut()` function divides the Age values into categories based on defined ranges: "Young" for ages from -Inf to 25, "Middle-aged" for ages between 25 and 40, and "Senior" for ages above 40. A new column called AgeGroup is created dynamically to store these categories.

3. Using the merged data, compute the average Score for each Age group created in Task 2. Explain the steps and provide the R code.

```
> # Calculate the average Score for each AgeGroup
> average_scores <- aggregate(Score ~ AgeGroup, data = merged_df, FUN = mean)
>
> # Display the result
> print(average_scores)
     AgeGroup Score
1       Young    85
2 Middle-aged    90
```

   The `aggregate()` function groups `merged_df` by `AgeGroup` and calculates the average score for each group using the formula `Score ~ AgeGroup`. The final result displays the average score for each age group.

4. Sort the merged data frame by Subject (ascending) and Score (descending). Write and explain the R code used.

```
> # Sort the data frame by Subject (ascending) and Score (descending)
> sorted_df <- merged_df[order(merged_df$Subject, -merged_df$Score), ]
>
> # Display the sorted data frame
> print(sorted_df)
  ID  Name Age Score Subject    AgeGroup
1  1 Alice  23    85    Math       Young
2  2   Bob  35    90 Science Middle-aged
```

The order() function sorts the data. The merged_df$Subject sorts by the Subject column in ascending order (default behavior). The -merged_df$Score sorts the Score column in descending order (the negative sign before Score reverses the sorting order). The data frame is then printed, showing the rows sorted by Subject and Score

5. Write an R function that accepts multiple data frames as input and combines them using rbind() after ensuring all have the same column names. Demonstrate the function with sample inputs.

```
> # Function to combine multiple data frames with the same column names
> combine_data_frames <- function(...) {
+     # Capture all the input data frames
+     data_frames <- list(...)
+
+     # Ensure that all data frames have the same column names
+     col_names <- lapply(data_frames, colnames)
+     if (length(unique(col_names)) != 1) {
+         stop("All data frames must have the same column names")
+     }
+
+     # Combine the data frames using rbind
+     combined_df <- do.call(rbind, data_frames)
+
+     return(combined_df)
+ }
>
> # Example data frames
> df1 <- data.frame(ID = 1:3, Name = c("Abegail", "Alexa", "Jethro"), Age = c(23, 35, 4
2))
> df2 <- data.frame(ID = 4:6, Name = c("Justine", "Nord", "Ely"), Age = c(28, 33, 38))
>
> # Demonstrate the function
> combined_df <- combine_data_frames(df1, df2)
>
> # Display the combined data frame
> print(combined_df)
  ID    Name Age
1  1 Abegail  23
2  2   Alexa  35
3  3  Jethro  42
4  4 Justine  28
5  5    Nord  33
6  6     Ely  38
```

The `combine_data_frames()` function takes multiple data frames as input using the ellipsis (`...`). It checks if all the data frames have the same column names by using `colnames()`. If the column names are different, the function stops and shows an error. If the column names match, it combines the data frames with `rbind()` using `do.call()`. The combined data frame is then returned.